REMARKS

Claims 6-7, 9-10 are pending in the above-referenced case. Claims 6 and 9 are amended. Claims 8 and 11 are cancelled without prejudice. Amendments to claims 6 and 9 are supported by, for example, paragraphs [0016]-[0019] and [0106]-[0114].

Claim 6 was rejected under 35 U.S.C. § 102(b) as being anticipated by Bishop and Dilger, "Checking for Race Conditions in File Access" (hereinafter, "Bishop").

Bishop discloses a static analyzer that analyzes C source code to detect file access race conditions. For a pair of sequential file access related system calls, the analyzer in Bishop determines whether the arguments to the system calls refer to the same file. (Bishop, page 10, 2nd paragraph.) The prototype analyzer in Bishop is a Perl script that "uses pattern matching over the source code to approximate generating and scanning a call dependency graph. It does no data flow analysis, but assumes that the file path name arguments are lexically identical in the system calls." (Bishop, page 11, 2nd paragraph, emphasis added.) Although Bishop contemplates the use of a call dependency graph and a data flow graph (Bishop, page 10, 2nd paragraph), it is silent on how these graphs may actually be used for determining whether the arguments to the system calls refer to the same file.

Bishop therefore does not disclose various features of amended claim 6. More specifically, Bishop does not disclose:

- "executing computer instructions to analyze the source code listing to create computer models of said control flow to indicate the run-time sequence in which routine calls will be invoked and to create computer models of said arguments for the routine calls **using a flow insensitive analysis**, wherein said control flow models include a control flow graph, and wherein each of **said models of arguments is stored in computer memory and specifies pre-determined characteristics about and a range of possible values for the corresponding_argument** as a result of said source code expressions"
- "executing computer instructions to use said computer models of said control flow in order to determine a run-time sequence of execution of a pair of routine calls by

traversing the control flow graph backwards, said pair of routine calls having a first routine call and second routine call in which execution of the first routine call precedes execution of said second routine call"

- "executing computer instructions to determine whether a second routine to be executed has a **second argument with a corresponding modeled range of possible of values that includes a reference to a file that is also within a corresponding modeled range of possible values for a first argument** of the first routine to be executed, so that a **possibility of the first and second arguments referring to the same file is determined even when said expression-references and operand-references to computer files for said first and said second arguments are lexically dissimilar**"

Bishop does not disclose "executing computer instructions to analyze the source code listing to create computer models of said control flow to indicate the run-time sequence in which routine calls will be invoked and to create computer models of said arguments for the routine calls **using a flow insensitive analysis**, wherein said control flow models include a control flow graph, and wherein each of **said models of arguments is stored in computer memory and specifies pre-determined characteristics about and a range of possible values for the corresponding argument** as a result of said source code expressions." (Emphasis added.) There is no teaching or suggestion in Bishop for creating models of arguments of routine calls that specify characteristics about and a range of possible values for the arguments. The prototype analyzer in Bishop has no use for such modeling because it simply determines whether two arguments are lexically identical. (Bishop, page 10, 3rd paragraph.) Furthermore, nowhere in Bishop is a flow insensitive analysis taught or suggested.

Bishop also does not disclose "executing computer instructions to use said computer models of said control flow in order to determine a run-time sequence of execution of a pair of routine calls by **traversing the control flow graph backwards**." (Emphasis added.) In rejecting claim 8, the Office Action concedes that Bishop "fails to expressly disclose traversing the control flow graph backwards in order to determine the sequential relationship among routine calls in the source code listing." (Office Action, page 4, last paragraph.) However, the Office Action combines Bishop with Jong-Deok Choi, et al., "Static Datarace Analysis for Multithreaded Object-Oriented Programs (hereinafter, "Choi"), and states that Choi page 13, col. 1, paragraphs 3-4 teaches the backwards traversing of a control flow graph. (Id.) The cited portion of Choi, however, only discloses a depth-first traversal of an interthread call graph (ICG), which is different from a backward traversal of a control flow graph. Therefore, even if Bishop and Choi can be combined, they do not teach or suggest "determin[ing] a run-time sequence of execution of a pair of routine calls by traversing the control flow graph backwards" as required by amended claim 6.

Moreover, Bishop does not disclose "executing computer instructions to determine whether a second routine to be executed has a **second argument with a corresponding modeled range of possible of values that includes a reference to a file that is also within a corresponding modeled range of possible values for a first argument** of the first routine to be executed, so that a **possibility of the first and second arguments referring to the same file is determined even when said expression-references and operand-references to computer files for said first and said second arguments are lexically dissimilar**." (Emphasis added.) As mentioned above, Bishop does not create models of arguments that specify pre-determined characteristics about and a range of possible values for the arguments. Therefore, Bishop cannot determine whether the range of possible values of the first argument and the range of possible values for the second argument include the same reference to a file. Again, the prototype analyzer in Bishop simply determines whether two arguments are lexically identical. (Bishop, page 10, 3rd paragraph.) In amended claim 6, however, ranges of possible values of the two arguments are examined for race condition determination, "so that a **possibility of the first and second arguments referring to the same file is determined even when said expression-references and operand-references to computer files for said first and said second arguments are lexically dissimilar**."

For at least the above reasons, Bishop and Choi do not anticipate amended claim 6. Therefore, applicants respectfully request that amended claim 6 be allowed.

Applicants further submit that independent claim 9, which is amended in a similar manner as claim 6, is allowable for at least the same reasons that claim 6 is allowable. In addition, dependent claims 7 and 10 are allowable for at least the same reasons that the independent claims are allowable.

In view of the above amendment, applicant believes the pending application is in condition for allowance.

The Commissioner is hereby authorized to charge any required fees to our Deposit Account No. 08-0219, under Order No. 0286685.00125US1 from which the undersigned is authorized to draw. Please apply any charges not covered, or any credits, to Deposit Account No. 08-0219.

Respectfully submitted,

Dated: February 28, 2008

/Peter M. Dichiara/
Peter M. Dichiara
Registration No.: 38,005
Attorney for Applicant(s)

Wilmer Cutler Pickering Hale and Dorr LLP
60 State Street
Boston, Massachusetts 02109
(617) 526-6000 (telephone)
(617) 526-5000 (facsimile)